May 1988 DTIC FILE COPY

UILU-ENG-88-2225
ACT-91

2

COORDINATED SCIENCE LABORATORY
College of Engineering
Applied Computation Theory

SCALING OF LINEAR PROGRAMS

Milind Ashok Sohoni

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS None	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-88-2225 (ACT-91)		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois	6b. OFFICE SYMBOL *(If applicable)* N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research	
6c. ADDRESS *(City, State, and ZIP Code)* 1101 W. Springfield Avenue Urbana, IL 61801		7b. ADDRESS *(City, State, and ZIP Code)* 800 N. Quincy St. Arlington, VA 22217	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-85-K-0570	
8c. ADDRESS *(City, State, and ZIP Code)* 800 N. Quincy St. Arlington, VA 22217		10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.	

11. TITLE *(Include Security Classification)*
Scaling of Linear Programs

12. PERSONAL AUTHOR(S) Sohoni, Milind Ashok

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Technical	FROM ______ TO ______	1988 May	36

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	linear programming, scaling algorithm, primal-dual algorithm, combinatorial optimization, weighted matching problem

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*
>The scaling technique has been used often in designing efficient algorithms for combinatorial optimization problems. This ~~paper~~ unifies problem-specific scaling approaches into a linear programming framework. Solution of linear programs by scaling involves successive solutions of what we call the *tuning problem*. This tuning problem arises when one transforms a solution for the previous scale into a solution for the next scale. ~~We show~~ that the tuning problem has a nice format under certain assumptions, and it is precisely this convenience which has led to fast scaling algorithms for many combinatorial problems. ~~We~~ also examines schemes that use a relaxation of complementary slackness, and ~~we~~ shows that one such scheme is equivalent to scaling. ~~We~~ proposes a generalization of an approximation algorithm by Gabow and Tarjan and discusses its application to the tuning problem. Finally, ~~we~~ discuss scaling of the shortest path problem and the weighted matching problem in ~~our~~ this linear programming framework.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE *(Include Area Code)* / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

SCALING OF LINEAR PROGRAMS

BY

MILIND ASHOK SOHONI

B.Tech., Indian Institute of Technology, Bombay, 1986

THESIS

Submitted in the partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1988

Urbana, Illinois

ABSTRACT

The scaling technique has been used often in designing efficient algorithms for combinatorial optimization problems. This paper unifies problem-specific scaling approaches into a linear programming framework. Solution of linear programs by scaling involves successive solutions of what we call the *tuning problem*. This tuning problem arises when one transforms a solution for the previous scale into a solution for the next scale. We show that the tuning problem has a nice format under certain assumptions, and it is precisely this convenience which has led to fast scaling algorithms for many combinatorial problems. We also examine schemes that use a relaxation of complementary slackness, and we show that one such scheme is equivalent to scaling. We propose a generalization of an approximation algorithm by Gabow and Tarjan and discuss its application to the tuning problem. Finally, we discuss scaling of the shortest path problem and the weighted matching problem in our linear programming framework.

ACKNOWLEDGEMENTS

I must thank my advisor Prof. M. C. Loui. Without his help and guidance this thesis would have taken much longer to complete. I also thank my roommate Sudin Bhat, and finally, The Daily Grind, Nature's Table and Shobha for much needed distraction.

This research was supported by the Office of Naval Research under Contract N00014-85-K-0570.

TABLE OF CONTENTS

INTRODUCTION

The scaling technique has been effective in designing efficient algorithms for weighted combinatorial problems that can be formulated as linear programs on integer polyhedra, such as shortest s-t path and weighted matchings. Edmonds and Karp [2] presented the first scaling algorithm for network flow problems. Later Gabow [3] [4] proposed scaling algorithms for a variety of weighted combinatorial problems. In this paper we unify the approaches of Gabow and others into a linear programming framework.

The typical scaling algorithm works as follows. Given the problem P with an integer vector of weights w, the algorithm recursively solves a problem P' that is identical to problem P but with w replaced by $\lfloor \frac{w}{2} \rfloor$. Then using the solution to P' it produces a solution to P. Thus if this step takes time $T(n)$, where n is a measure of the size of the problem, then the algorithm solves P in time $T(n) \log W$, where W is the largest weight in the problem.

Scaling may be seen as a method of "combinatorializing" an optimization problem. The technique is not as powerful as the Primal-Dual algorithm: While the Primal-Dual algorithm leads to a series of $\{0,1\}$ weighted problems, the scaling algorithm leads to $\{0,1,...,k\}$ weighted problems. Nevertheless, scaling is sufficiently powerful to yield efficient algorithms with sophisticated data structures.

In Section 1 we first present a linear programming (LP) interpretation of scaling. We discuss the operation of the scaling algorithm on an integer LP, and we formulate the problem of transforming a solution to P into a solution to P'. We call this transformation problem the *tuning problem*. We obtain results on general LPs that have been used in many specific scaling algorithms. Using these results, we simplify the scaling algorithm of Gabow for weighted general matching [4]. In Section 2 we present an analysis of Bertsekas's approximate complementary slackness technique [1] and show its equivalence to scaling. We also examine successive approximation methods for solving the tuning problem. In Section 3 we show how specific scaling algorithms can be cast in the general framework of Section 1. An alternate formulation of the general weighted matching algorithm is presented.

CHAPTER 1

SCALING OF LINEAR PROGRAMS

Let A be a matrix and b, c be vectors. Consider the linear program

$$\begin{aligned} &\min cx \\ &Ax=b \\ &x\geq 0 \end{aligned} \tag{1.1A}$$

Let us assume that we have solved (1.1A) above. We shall examine how this has helped us in solving the perturbed LP (1.2A) below.

$$\begin{aligned} &\min cx+dx \\ &Ax=b \\ &x\geq 0 \end{aligned} \tag{1.2A}$$

If the vector d has special properties, then we shall see that solving (1.1A) has made the task of solving (1.2A) much easier. To see this, let us consider the duals of (1.1A) and (1.2A), which are (1.1B) and (1.2B) respectively.

$$\begin{aligned} &\max yb \\ &yA\leq c \end{aligned} \tag{1.1B}$$

$$\begin{aligned} &\max yb \\ &yA\leq c+d \end{aligned} \tag{1.2B}$$

Let x_0 be an optimal solution to (1.1A) and y_0 an optimal solution to its dual (1.1B). Define $e=c-y_0A$, and call e the *slack vector* in (1.1B). By (1.1B), $e\geq 0$. Note that by complementary slackness $ex_0=0$. Now substitute y_0+y for y in (1.2B) and simplify to obtain (1.3B) below.

$$\max yb$$

$$yA \leq d+e \tag{1.3B}$$

The dual of (1.3B) is (1.3A) below.

$$\min\ (d+e)x$$

$$Ax=b \tag{1.3A}$$

$$x \geq 0$$

Let us examine x_0 as a tentative solution to (1.3A). Since $ex_0=0$, it follows that $(d+e)x_0=dx_0$. At this point let us assume the following:

(i) d is non-negative, integral.

(ii) A, b, and c are integral.

(iii) $Ax=b$ defines a polyhedron with integral vertices.

(iv) e is integral.

Many optimization problems satisfy the above assumptions.

Since d and e are non-negative, the optimal value of (1.3A) is non-negative. As the cost of x_0 is dx_0, x_0 is within dx_0 of the optimal value. Let (x_1,y_1) be an optimal pair of solutions to (1.3A),(1.3B). For each j, if $(x_1)_j>0$, then by complementary slackness of (x_1,y_1) we have:

$$(y_1A)_j=(d+e)_j.$$

Consequently $((y_0+y_1)A)_j=(y_0A)_j+(y_1A)_j=(c-e)_j+(d+e)_j=(c+d)_j$. Thus we see that (x_1,y_0+y_1) exhibits complementary slackness with respect to (1.2A),(1.2B) implying that the pair is optimal for (1.2A),(1.2B). Therefore a solution to (1.3A) itself is a solution to (1.2A). Also note that since $y_0b=cx_0$, y_0 is a feasible solution of (1.2B) and is within dx_0 of the optimum.

Let us examine the problem P defined below.

$$\min gx \quad (g \text{ is integral})$$
$$Ax=b \tag{1.4A}$$
$$x\geq 0$$

Assume (ii) and (iii) hold for P. Define $d=g \bmod 2$ and $c=g-d$. Hence $\lfloor \frac{g}{2} \rfloor = \frac{c}{2}$. By definition, d is 0-1 and c is a vector of even integers. The scaling algorithm first solves a problem P' by changing g to $\lfloor \frac{g}{2} \rfloor$:

$$\min \frac{c}{2}x \quad (\frac{c}{2} \text{ is integral})$$
$$Ax=b \tag{1.5A}$$
$$x\geq 0$$

Note that (1.5A) satisfies conditions (ii) and (iii). Suppose we solve (1.5A) recursively. If (x_1,y_1) is an optimal solution to (1.5A) and its dual, then $(x_1,2y_1)$ is an optimal solution to (1.1A),(1.1B). Therefore the slack $e=c-2y_1A$ in (1.1B) is twice the slack $\frac{c}{2}-y_1A$ in the dual of (1.5A). Thus if the entries in y_1 are integral multiples of $\frac{1}{2}$ (i.e., half-integers), then e is integral, and (1.3A) satisfies conditions (i)-(iv). The scaling algorithm then has to solve (1.3A),(1.3B) to produce a solution of (1.4A). Thus we have broken the problem P into (1.1A) and (1.3A). Problem (1.1A) is equivalent to (1.5A), and the largest component in the cost vector of (1.5A) is at most half of the largest component in (1.4A). Working under assumptions (i)-(iv), we shall examine (1.3A),(1.3B) closely.

Let x_0 be a solution to (1.1A) and define $k=dx_0$. Since $Ax=b$ has integral solutions, k is an integer. Then "trim" the vector $d+e$ in (1.3A) as follows. Replace all entries in $d+e$ that are greater than k by $k+1$. This trimming does not change the solutions to (1.3A) because we know that if the j th component of $d+e$ is larger than k, then the corresponding x_j cannot be positive in the optimal solution because the optimal value of (1.3A) is strictly *less* than $k+1$. Alternatively, delete the columns in A corresponding to the entries greater than k in $d+e$ to get a reduced (1.3A). Thus if e is integral, then to solve (1.2A) we just have to solve the problem below, which we call the *tuning problem*.

$$\begin{gathered}\min\,(d+e)x \quad (d+e \text{ is trimmed}) \\ Ax=b \\ x\geq 0\end{gathered} \tag{1.6A}$$

We call the vector $d+e$ the *offset*. Since the vector $d+e$ can take values from the finite set $\{0,\ldots,k+1\}$ we call the problem semi-combinatorial [7]. Efficient data structures may then be used solve the new (1.6A).

In summary, we see that the problem (1.4A) can be broken into two problems that have the same format as (1.4A), viz. (1.5A) and (1.6A). Problem (1.5A) may be solved recursively. Problem (1.6A), the tuning problem, depends on (1.5A) only through the slack vector for the solution to (1.5A). The solution to (1.6A) is also the solution to (1.4A). The optimal value of (1.6A) is bounded by dx_0 where x_0 is a non-negative solution to $Ax=b$.

For many combinatorial problems, although A, b, and c are integral, the components of the optimal solutions of (1.1B) are not integral but are integral multiples of $\frac{1}{l}$, where l is an integer. In Section 3 we shall examine specific LPs with $l=2$ and $l=4$. In this situation, instead of scaling by 2, scale by the factor l. That is, given the problem:

$$\begin{gathered}\min gx \\ Ax=b \\ x\geq 0\end{gathered} \tag{1.4A}$$

put $d=g \bmod l$ and $c=g-d$. Now (1.1A) is equivalent to the problem (1.5C) below, which is again solved recursively.

$$\min \left(\frac{c}{l}\right)x$$
$$Ax=b \tag{1.5C}$$
$$x\geq 0$$

As $\frac{c}{l}$ too is integral, A, b, and the cost vector $\frac{c}{l}$ are all integral. By hypothesis, the optimal values for the dual variables of (1.5C) are multiples of $\frac{1}{l}$. By multiplying the dual variables of (1.5C) by l we obtain an optimal solution for (1.1B), which must be integral. Then the slack vector $e=c-y_0A$ is integral, thus ensuring that the tuning problem satisfies assumptions (i), (ii), and (iii), above.

We now show that the tuning problem (1.6A) itself cannot be solved by scaling. For let $d+e=c_1+d_1$ where the components of d_1 are in $\{0,1\}$ (respectively in $\{0,...,l-1\}$) and c_1 is even (respectively divisible by l). By complementary slackness, if $(x_0)_j>0$, then $(c_1)_j=l\left\lfloor\frac{(d+e)_j}{l}\right\rfloor=l\left\lfloor\frac{(d)_j}{l}\right\rfloor=0$, making $c_1x_0=0$. Hence the new scaled down problem (1.5C) for (1.6A) has zero as its optimal value, x_0 itself being a solution.

We have discussed scaling on LPs involving minimization of a cost function. If the given LP is a maximizing problem, then a slight modification of scaling may be used. Examine problems (1.7A) and its dual (1.7B), and (1.8), (1.9) below.

$$\max wx$$
$$Ax=b \tag{1.7A}$$
$$x\geq 0$$

$$\min yb$$
$$yA\geq w \tag{1.7B}$$

$$\max \left\lceil\frac{w}{2}\right\rceil x$$

$$Ax=b \tag{1.8}$$
$$x\geq 0$$

$$\max \left\lfloor \frac{w}{2} \right\rfloor x$$
$$Ax=b \tag{1.9}$$
$$x\geq 0$$

For convenience we refer to LP (1.7A) as $P(w)$, (1.7B) as $DP(w)$, (1.8) as $P(\lceil \frac{w}{2} \rceil)$, and (1.9) as $P(\lfloor \frac{w}{2} \rfloor)$. We claim that the scaled down problem should be taken as $P(\lceil \frac{w}{2} \rceil)$ instead of $P(\lfloor \frac{w}{2} \rfloor)$ for the following two reasons:

(a) If P is a maximizing problem, then its dual DP is a minimizing problem. Thus if y_0 is a solution to $DP(\lfloor \frac{w}{2} \rfloor)$, then $2y_0$ need not even be feasible for $DP(w)$, whereas if y_1 is a solution to $DP(\lceil \frac{w}{2} \rceil)$, then $2y_1$ is certainly feasible for $DP(w)$.

(b) The slack vector $e=w-y_0A$ is non-positive, hence it is useful to have d (as in assumption (i) above) non-positive. Observe that in the case of scaling on a minimization problem, the costs of intermediate dual solutions at succesive scales increase monotonically. In the case of a maximization problem, choosing $P(\lceil \frac{w}{2} \rceil)$ as the scaled down problem makes the cost of intermediate dual solutions decrease monotonically .

We can apply the observations of this section to simplify Gabow's algorithm for maximum cost weighted general matching [4]. Gabow uses $P(\lfloor \frac{w}{2} \rfloor)$ as the scaled down problem. The dual variables then require a slight adjustment to retain feasibility (see (a) above). His algorithm and its correctness proof may be simplified by choosing $P(\lceil \frac{w}{2} \rceil)$ as the scaled down problem. This follows from (a) and (b) above. Furthermore, as $l=4$ (i.e., the dual variables are multiples of $\frac{1}{4}$) it is more convenient to scale by 4 than by 2. Consequently, for a simpler algorithm, $P(\lceil \frac{w}{4} \rceil)$ may be chosen as the scaled down problem.

CHAPTER 2

APPROXIMATE COMPLEMENTARY SLACKNESS

2.1 Bertsekas's Approximate Complementary Slackness. Consider the linear program (2.1A) and its dual (2.1B).

$$\begin{aligned} &\max cx \\ &Ax=b \\ &x\geq 0 \end{aligned} \tag{2.1A}$$

$$\begin{aligned} &\min yb \\ &yA\leq c \end{aligned} \tag{2.1B}$$

Let x^0 be feasible in (2.1A) and y^0 be feasible in (2.1B). Define the slack vector $e^0=c-y^0A$. By complementary slackness, the pair (x^0,y^0) is optimal in (2.1A),(2.1B) respectively if and only if $e^0x^0=0$. Restated, (x^0,y^0) is optimal if for all j, whenever $e_j{}^0>0$, $x_j{}^0=0$.

Bertsekas [1] introduced the notion of "approximate" complementary slackness for the min-cost flow problem. Call the pair of feasible solutions (x^1,y^1) ε*–optimal* if $e_j{}^1\geq\varepsilon$ implies $x_j{}^1=0$ for all j. Note that a 0-optimal solution is an optimal solution to (2.1A),(2.1B) because it satisfies the complementary slackness conditions.

Approximate complementary slackness has been used to design efficient algorithms for network problems [5,6]. For a given problem, it is first proved that there exists an $\varepsilon_0>0$ such that if a solution is ε_0-optimal, then it is 0-optimal. This ε_0 is specific to each problem and depends on the values the dual variables can take. The typical algorithm first starts with an initial guess (x^0,y^0). If (x^0,y^0) is 0-optimal, then the algorithm stops since (x^0,y^0) are optimal. If the solution is not 0-optimal, then there *must* exist an ε such that x^0,y^0 is ε-

optimal: For example, we can take $\varepsilon = \max_j \{c_j - (y^0A)_j\}$. This ε-optimal solution is used to generate an $\frac{\varepsilon}{2}$-optimal solution, which is then used to generate an $\frac{\varepsilon}{4}$-optimal solution, and so on, until finally we arrive at an ε_0-optimal solution. This process may be alternatively viewed as follows. Consider the scaled version of (2.1A), where the scaling factor is 2^k, i.e., the new cost function is $\lfloor \frac{c}{2^k} \rfloor$ as shown below.

$$\min \lfloor \frac{c}{2^k} \rfloor x$$

$$Ax = b \tag{2.2A}$$

$$x \geq 0$$

Let (x_k, y_k) be an optimal solution to (2.2A) and its dual. Then $(x_k, 2^k y_k)$ is a 2^k-optimal solution to (2.1A) because (x_k, y_k) is an optimal solution to (2.2A). If there existed a j such that $(c - 2^k y_k A)_j > 2^k$ but $(x_k)_j > 0$, then $(\lfloor \frac{c}{2^k} \rfloor - y_k A)_j > 0$ but $(x_k)_j > 0$, contradicting complementary slackness of (x_k, y_k) in (2.2A). Thus, one way to arrive at a 2^{k-1}-optimal solution would be to solve the tuning problem for (2.2A) to obtain an optimal solution for the LP with cost vector of $\lfloor \frac{c}{2^{k-1}} \rfloor$. This process is continued until it finally arrives at a k_0 such that $2^{k_0} < \varepsilon_0$ specified previously.

If problem (2.1) satisfies assumtions (ii) and (iii) of Section 1, and the dual variables are multiples of $\frac{1}{l}$ (l is an integer), then the components of the slack vector are integral multiples of $\frac{1}{l}$. Therefore the value of ε_0 is precisely $\frac{1}{l}$ where l is also the scaling factor discussed in the Section 1. For the bipartite matching LP $\varepsilon_0 = \frac{1}{2}$, and for the general matching LP $\varepsilon_0 = \frac{1}{4}$.

2.2 k-feasibility and k-optimality. Tarjan and Gabow [5] applied an alternate formulation of approximate complementary slackness to the bipartite matching LP. We shall attempt to generalize and give linear programming interpretations of their formulation. For the LP (2.1A) and a non-negative vector k, the pair of vectors (x_k, y_k) is called *k-feasible* if

(i) $(x_k)_j > 0$ implies $(y_k A)_j = c_j$

(ii) $(x_k)_j = 0$ implies $(y_k A)_j \leq c_j + k_j$

Define the vector x_k to be *k-optimal* if it is k-feasible and x_k is feasible in its LP (2.1A), i.e., $Ax_k = b$. But note that the vectors of a k-feasible pair need not even be feasible in their respective LPs.

Let us first analyse how close a k-optimal solution x_k is to the optimal value for (2.1A). First note that (x_k, y_k) is feasible in the following LP and its dual.

$$\begin{aligned} \min\ & (c+k)x \\ & Ax = b \\ & x \geq 0 \end{aligned} \qquad (2.3)$$

$$\begin{aligned} \max\ & yb \\ & yA \leq c+k \end{aligned}$$

Note that

$$(c+k)x_k - y_k b = (c+k)x_k - (y_k A)x_k = (c+k-y_k A)x_k .$$

By k-optimality of (x_k, y_k), if $(x_k)_j > 0$ then $(y_k A)_j = c_j$. Hence, for all j, $(c+k-y_k A)_j (x_k)_j \leq k_j (x_k)_j$. Consequently, $(c+k)x_k - y_k b \leq kx_k$. This implies that x_k is within kx_k of the optimal value for (2.3). And if x_0 is an optimal solution to (2.1A), then the optimal value for (2.3) is at most kx_0 away from that of (2.1A). Thus x_k is within $k(x_0 + x_k)$ of the optimal value of (2.1A) and x_k is feasible in (2.1A). Define $||x||$ to be the l_1 norm of vector x. If all the solutions of $Ax=b$ are bounded, i.e., $||x|| \leq n$ for all solutions of $Ax=b$, then x_k

is within $m = n \,||k||$ of the optimal value for (2.1A).

The algorithm to achieve this k-optimal solution proceeds as follows: First the cost function c in (2.1A) is multiplied by $m+1$. This ensures that any k-optimal solution to this modified (2.1A) is also *optimal* for it. A starting k-feasible solution x_0 is then selected. Through successive refinements this k-feasible solution is made k-optimal. Note that k-feasibility is more relaxed than complementary slackness (or 0-optimality) and is potentially easier to maintain.

Now what remains is the specification of the refinement procedure. This refinement procedure should start with a k-feasible solution (x_1,y_1) and produce another k-feasible solution that is either k-optimal or "better" than (x_1,y_1).

Here, we state a refinement procedure that is a modified relaxation of the Primal-Dual algorithm. It may be interpreted as a generalization of procedure *Match* of [5]. Let us first review the Primal-Dual algorithm [7]. Given an LP P, its dual DP, and vector π feasible in DP, the Primal-Dual algorithm first constructs the restricted primal RP and its dual DRP. Optimal solutions x',π' for RP and DRP are found, and π is upgraded to $\pi+\theta\pi'$, where θ is chosen appropriately. Since (x',π') are optimal for RP and DRP, x' and $\pi+\theta\pi'$ satisfy the complementary slackness condition. Note that x' itself need not be feasible in P. The efficiency of the Primal-Dual algorithm lies in this inheritance of complementary slackness, as at the beginning of the next iteration we have a valid candidate solution to start with.

For a given LP P as in (2.1A) and a vector x, define the *k-perturbed* LP $P(x)$ with respect to x to be the dual of the following LP:

$$\max\ yb$$

$$yA \le c + \delta k$$

where

$$\delta_j=0 \text{ if } x_j>0$$

$$\delta_j=1 \text{ if } x_j=0$$

Note that if (x_k, y_k) is k-feasible for P and its dual, then (x_k, y_k) satisfies complementary slackness conditions with respect to $P(x)$. Also note that while x_k need not be feasible in the primal, y_k is always feasible in the dual of $P(x)$.

We now state procedure *Refine* below.

Procedure *Refine*

{Input : (x_0, y_0) which is k-feasible in P and its dual

Output : (x_1, y_1) k-feasible or k-optimal. }

Step 0. Find the restricted primal RP with respect to $P(x_0)$.

Step 1. In RP find a k-optimal solution, if one exists. Otherwise find a "better" k-feasible solution x_1. Modify the dual variables from y_0 to y_1 such that x_1 is optimal in the RP with respect to $P(x_1)$.

Step 2. Update the dual variables as in the Primal-Dual algorithm until a better k-feasible solution can be found. □

In Step 1, if a k-optimal solution is found, then this solution is the desired k-optimal solution. If not, then the dual variable modifications ensure that (x_1, y_1) maintain complementary slackness in the RP of $P(x_1)$. This ensures the validity of Step 2, because complementary slackness of (x_1, y_1) after Step 1 is required for Step 2 to work. Step 1 involves finding a vector y_1 that is feasible in the dual of $P(x_1)$ and maintains complementary slackness with x_1. This process may be made more efficient if a "better" x_1 is found in a systematic way. Step 2 essentially looks for an optimal solution to $P(x_1)$, and therefore unless x_1 is k-optimal, a better k-feasible solution can always be found. Thus at the end of Step 2, we are guranteed that a better solution can indeed be found. The computational convenience of the algorithm is discussed in the Section 2.3 .

We shall compare our procedure *Refine* witn procedure *Match* in [5], which solves the weighted bipartite matching problem for the graph $G(V_1,V_2,E)$. Define $\mathbf{1}=(1,1,...,1)$. Procedure *Match* finds a 1-optimal matching, thus $k=(1,1...,1)$ for *Match*. Let $y(v)$ be the dual variable associated with vertex v and $c(v,w)$ be the cost of edge (v,w). If M is a matching (not neccesarily complete), then an edge (v,w) is called ***eligible*** if

$$y(v)+y(w)=c(v,w)+\{\text{if } (v,w)\in M \text{ then } 0 \text{ else } 1\ \}.$$

Procedure *Match*

Step 1. Find a maximal set A of vertex disjoint augmenting paths of eligible edges. For each path $P \in A$, augment the matching along P, and for each vertex $w\in V_1\cap$P, decrease $y(w)$ by 1. (This makes the new matching M' 1-feasible.) If the new matching M' is complete, halt.

Step 2. Do a Hungarian search to adjust the duals and find an augmenting path of eligible edges. □

Note that the set of eligible edges generates precisely the RP of $P(M)$. Step 1 of *Match* is equivalent to Step 1 of *Refine*. Finding a *maximal* set of augmenting paths and decrementing partially the dual variables ensure that the new matching and the dual variables maintain complementary slackness, *and* that the new matching is optimal in the RP with respect to the new duals. This ensures the validity of the Hungarian search of Step 2. Let us look at Step 1 of *Match* more closely. In particular, we shall examine the notion of "better" k-feasible solution (Step 1 of *Refine*) in relation to *Match*. First note that any intermediate 1-feasible solution x_0 (in *Match*) satisfies $Ax_0\leq b$, where A,b are from the bipartite matching LP primal, because x_0 always represents a matching, though not neccesarily a complete matching. Thus x_1, another 1-feasible solution, may be called better than x_0 if the slacks satisfy $||b-Ax_0|| > ||b-Ax_1||$. Since the matrix A for the bipartite matching LP is totally unimodular, for integral intermediate k-feasible solutions (x_1,y_1), $||b-Ax_1||$ is integral. Thus $||b-Ax_1||$ is a good measure of x_1, because as long as our intermediate solutions remain integral, the measure is integral and decreases at every execution of *Refine*. We call this measure the *slack* measure of goodness. Consequently if in Step 1 of *Refine* we use this notion of betterness, then finding a better k-feasible solution is equivalent to finding augmenting paths, flows etc. for standard LPs (matching, network flows etc.).

2.3 Procedure *Refine* and The Tuning Problem. In this subsection we shall present a subjective analysis of the application of *Refine* to the tuning problem arising from scaling. We noted in Section 1 that the tuning problem itself cannot be simplified by further scaling. We also noted, however, that the optimal value of the tuning problem is bounded by $||x_*||$, where x_* is the solution to the previous scale (assuming we are scaling by 2). Let us apply *Refine*, with the slack measure, to the tuning problem in order to find a k-feasible solution for it. As the tuning problem is the core of the scaling algorithm, we need to estimate how efficient *Refine* is on the tuning problem.

Let us analyse the number of iterations required for *Refine* to arrive at a k-feasible solution. First, in Step 1 we see that the tentative k-feasible solution improves in slack, i.e., $||b-Ax_1||>||b-Ax_0||$, while in Step 2 we see that there is an improvement in the dual function value. Step 2 of *Refine* is identical to that of the Primal-Dual algorithm, and hence this improvement is guranteed. Further, we also know that the optimal value to the problem cannot exceed $||x_*||$. Then if the dual variable increments are directly related to the slack measure $||b-Ax_1||$ and the dual variable modification of Step 1 does not substantially decrement the dual variables, then *Refine* converges quickly to a k-feasible solution. In other words, if Step 1 is slow in finding k-feasible solutions, then as the slack measures are large, Step 2 increments dual variables rapidly, and vice-versa. But the cost of the dual solution is bounded by $||x_*||$ because the optimum value of the tuning problem is bounded $||x_*||$. Hence, at every iteration of *Refine*, either the slack measure decreases or the cost of the dual solution approaches its upper bound. This puts an upper bound on the number of iterations of *Refine* required. The situation of the dual variable update being directly related to the slack measure is common to most of the network flow LPs. In procedure *Match*, for example, at each execution of Step 2, the increase in the cost of the dual variables is $\frac{1}{2} \cdot n_0$, where n_0 is the number of unmatched vertices at that stage. But $n_0 = ||b - Ax_1||$, the measure of x_1 in the bipartite matching LP. Thus the dual variable update is indeed directly related to the goodness measure.

CHAPTER 3

APPLICATIONS

In the previous sections we have established some facts about scaling. This section shows how these facts have been used by other researchers in earlier papers.

3.1 Shortest Paths with Non-Negative Edge Lengths. For a directed graph $G(V,E)$ and a non-negative integer length l_{ij} associated with each arc (i,j) in E, the shortest path problem is to find directed paths of minimum length from a distinguished source vertex s to all other vertices of G. Let $m=|E|$. We shall present Gabow's algorithm [3] for the problem and show that it is an instance of the more general algorithm of scaling of linear programs. Gabow first defines a "near-optimum" solution as a vector p, with one entry p_i for each vertex i, such that:

(i) $p_s=0$.

(ii) p *dominates* edge lengths, that is, for any edge ij, $p_i+l_{ij}\geq p_j$ where $l_{ij}\geq 0$.

(iii) The length of the shortest path from s to i is between p_i and p_i+m.

Thus p_i is an estimate of the distance from s to i.

Let P be our original problem and let P' be the scaled down problem (weights l_{ij} replaced by $\lfloor \frac{l_{ij}}{2} \rfloor$). Gabow then proceeds as follows: Recursively obtain the optimal distance vector p_i' for P'. Define $p=2p'$. Then p is a near-optimum distance vector for P. Using this near-optimum p_i, compute modified edge lengths, $l_{ij}'=l_{ij}-(p_j-p_i)$. Compute shortest paths on G with the modified edge lengths l_{ij}' using Dijkstra's algorithm. The shortest paths on this modified graph are the shortest paths for the problem P.

The efficiency of the algorithm hinges on the use of an array $Q[0,\dots,m]$ as the priority queue in Dijkstra's algorithm. $Q(k)$ stores vertices at *tentative* distance k. From property (iii) above of a near-optimal solution, the distances in the modified graph are at most m, and this implies that Q suffices as the priority queue. Since solving this modified problem takes $O(m)$ time, P can be solved in $O(m \log N)$ time, where N is the largest weight in P, which is the largest edge cost.

Now we frame the shortest path problem as an LP and interpret Gabow's algorithm in relation to the results from Section 1. The shortest path LP is stated below:

$$\min \sum_{i,j} l_{ij} f_{ij}$$

$$Af = b \qquad (3.1A)$$

$$f \geq 0$$

where $b=[-(n-1),1,1,\dots,1]^T$, $n=|V|$, and A is the $n \times m$ edge incidence matrix with the first row representing the source s. Its dual is:

$$\max \sum_{i=2}^{n} p_i - (n-1)p_1$$

$$pA \leq l \qquad (3.1B)$$

Note that the LP in (3.1A) satisfies assumptions (ii) and (iii) of Section 1. From the dual LP, we see that the p_i's in Gabow's algorithm are the dual variables of the shortest path LP for P'. The modified problem of Gabow is then precisely the tuning problem (1.6A) of Section 1 because

$$l'_{ij} = l_{ij} - (p_j - p_i) = 2\left\lfloor \frac{l_{ij}}{2} \right\rfloor + l_{ij} \bmod 2 - (p_j - p_i)$$

which is precisely $d_{ij} + e_{ij}$, the offset defined in Section 1. Furthermore the solution to the tuning problem (1.6A) is a solution to P(1.4A), ensuring that the shortest paths on the modified graph are the shortest paths on the original graph, and thus proving the correctness of Gabow's algorithm.

In this case e is integral, and the trimming of $d+e$ suggested in Section 1 guarantees the effective use of Q as the priority queue. The above example illustrates that the tuning problem may sometimes be efficiently solved using data structures that utilize the integrality and boundedness of the offset.

3.2 Weighted Bipartite Matchings. We study the scaling algorithm of Gabow for weighted bipartite matching [3]. This algorithm shows how scaling applies to the Primal-Dual algorithm for solving linear programs.

Let $G(V_1,V_2,E)$ be a bipartite graph, where V_1 and V_2 are vertex sets of the same size and E is the edge set. Given an integer cost function c_{ij} on E, the problem is to a find a complete matching in G of mimimum cost. Let $n=|V_1|=|V_2|$.

The LP of the weighted bipartite matching problem is given below.

$$\min \sum_{i,j} c_{ij}x_{ij}$$

$$\sum_{j=1}^{n} x_{ij}=1 \qquad i=1,\ldots,n \tag{3.2A}$$

$$\sum_{i=1}^{n} x_{ij}=1 \qquad j=1,\ldots,n$$

$$x_{ij}\geq 0 \quad \text{for all } i,j$$

The dual of this problem is:

$$\max \sum_{i=1}^{n}\alpha_i+\sum_{j=1}^{n}\beta_j$$

$$\alpha_i+\beta_j\leq c_{ij} \qquad \text{for all } i=1,\ldots,n,\ j=1,\ldots,n. \tag{3.2B}$$

We note that for bipartite matchings :

(a) The bipartite matching LP satisfies assumptions (ii) and (iii) of Section 1.

(b) In the Primal-Dual algorithm, at every step the value of the optimizing function increses by *at least* $\frac{1}{2}$.

(c) If all the edge costs are integers, then the dual variables are multiples of $\frac{1}{2}$.

Points (b) and (c) above may be proved as follows. Let us define the parity of a number as 1 if it is an odd multiple of $\frac{1}{2}$, 0 if it is an even multiple, and undefined otherwise.

Claim. At all times during the execution of the Primal-Dual algorithm on (3.2A) and (3.2B), all the dual variables have the same parity.

Proof. By induction on the number of steps, or dual variable modifications. The initial values of the dual variables are:

$$\alpha_i = 0 \text{ for } i=1..n.$$

$$\beta_j = \min_{1\le i\le n} c_{ij}$$

As all of $\{c_{ij}\}$ are integral, all of $\{\alpha_i\}$ and $\{\beta_j\}$ are of parity 0.

Recall that to modify the dual variables, first $\theta = \frac{1}{2}\min_{ij}(c_{ij} - \alpha_i - \beta_j)$ is calculated over a subset of the edges. Thus if all the dual variables have the same parity, then the parity of $\alpha_i + \beta_j$ is 0, and the parity of θ is well defined. At every step the dual variables are modified as follows: $\alpha_i \leftarrow \alpha_i \pm \theta$, $\beta_j \leftarrow \beta_j \pm \theta$. All the modified dual variables will then have the same parity. Thus at all times the dual variables are half-integers and (c) above is true. □

To establish (b), the increase in the dual objective function is $\theta\pi' b$ [7, Section 5.2]), where π' is a solution to the dual restricted problem DRP. In our case π' is integral and $b' = [1,1 \ldots 1]^T$. Thus the increase must be an integral multiple of θ, which is at least $\frac{1}{2}$.

Point (c) above suggests scaling by 2. In this subsection and the next, we shall use y (with or without subscripts) to refer to all the dual variables collectively. If (x_1, y_1) is an optimal solution to (1.5A) and its dual, then $(x_1, 2y_1)$ is a feasible solution to (1.4A),(1.4B). Further, from (c) we see that $(x_1, 2y_1)$ is integral. Therefore the vector e is integral. Then the problem (1.6A) is another bipartite matching problem with integral edge costs. Furthermore the optimal value of this problem is at most $dx_1=\frac{n}{2}$. Now there are two options. Either we could start with $2y_1$ as an initial choice for (1.2B) and work toward the optimum, or we could solve (1.6A), in both cases using the Primal-Dual algorithm. Because of (a), in either case there can be $O(n)$ dual variable updates (steps). Gabow obtained a fast algorithm that takes the first option. That there are only $O(n)$ steps is crucial in the analysis.

3.3 Weighted General Matchings. Given a complete graph $G(V,E)$ and an integer cost function c_{ij} on E, the problem is to find a complete matching of minimum cost. Let $n=|V|$.

Consider the general matching LP and its dual below [7, Section 11.3]. Suppose there are N odd-sets $S_1,\dots,S_N$, and, for each k let $s_k=\frac{|S_k|-1}{2}$.

$$\min \sum_{i,j} c_{ij}x_{ij}$$

$$\sum_{j=1}^{n} x_{ij}=1 \qquad i=1,\dots,n \tag{3.3A}$$

$$\sum_{i,j\in S_k} x_{ij} + w_k = s_k \qquad \text{for each odd-set } S_k \ k=1,\dots,N.$$

$$x_{ij}\ge 0 \qquad w_k\ge 0$$

$$\max \sum_{i=1}^{n}\alpha_i + \sum_{k=1}^{N} s_k\gamma_k \tag{3.3B}$$

$$\alpha_i+\alpha_j+\sum_{i,j\in S_k}\gamma_k\le c_{ij} \qquad \text{for all } i,j\le n$$

$$\gamma_k\le 0 \qquad \text{for all } k\le N$$

We note that

(a) The general matching LP satisfies assumptions (ii) and (iii) of Section 1.

(b) At every step in the Primal-Dual algorithm the value of the objective function increases by at least $\frac{1}{4}$.

(c) If all edge costs are integers, then the dual variables are multiples of $\frac{1}{4}$ (i.e., quarter-integers).

Statements (b) and (c) above may be proved as follows. Let us define the *parity* of a number as 0 if it is an even multiple of $\frac{1}{4}$, 1 if it is an odd multiple, and undefined otherwise. The parity of a set is defined only if all the elements of the set have the same parity, and then the parity of the set is defined to be the parity of any of its elements.

Let us partition V into two sets, M and X, depending on the current matching on the graph: M contains the matched vertices and X the exposed ones. Initially, in the Primal-Dual algorithm, $V=X$, and the algorithm proceeds by successively reducing X. Thus once a vertex has been matched it will remain matched (although its mate may change).

An edge (i,j) of G is called *tight* with respect to the current dual variables if $\alpha_i+\alpha_j+\sum_{i,j\in S_k}\gamma_k=c_{ij}$. The algorithm constructs the *current* or *tight* graph G_c, which is the graph induced by the tight edges at that stage [7, Section 11.3] and updates the previous matching. In this process some vertices of X may become matched. For this to happen, these vertices must be connected to some vertices in M by edges in G_c (i.e., tight edges).

Claim. Let M, X be a partition of V as defined above. Then

(i) All dual variables α_i for $i\in M$ have the same parity.

(ii) All α_i for $i\in X$ have parity 0.

(iii) All the odd-set variables γ_k, have parity 0.

Note that *parity* (M) need not be the same as *parity* (X).

Proof. By induction on the number of steps or dual variable modifications. The initial solution for the dual variables is:

$$\alpha_i = \frac{1}{2} \min_j c_{ij},\ \gamma_k = 0 \text{ for all } k.$$

Since initially $X=V$ and every c_{ij} is an integer, our claim is true for the initial solution. Recall that the dual variables are modified according to the following rule:

$\delta_1 = \frac{1}{2}\min\{c_{ij}-\alpha_i-\alpha_j : (i,j)\in T\subseteq M\times M\}$ (for the exact definition of T see [7, Section 11.3])

$\delta_2 = \min\{c_{ij}-\alpha_i-\alpha_j : i\in M, j\in X\}$

$\delta_3 = \min(\frac{-\gamma_k}{2})$

$\theta = \min(\delta_1, \delta_2, \delta_3)$

$\alpha_i \leftarrow \alpha_i \pm \theta$ for $i\in M$

$\alpha_i \leftarrow \alpha_i$ otherwise (i.e., $i\in X$)

$\gamma_k \leftarrow \gamma_k \pm 2\theta$.

Let M_1, X_1 be the partition before a step and M_2, X_2 after the step. By construction, $M_1 \subseteq M_2$. Assume that the dual variables satisfy the claim before the step. Then we show that the dual variables satisfy the claim after the step.

If $M_1 = M_2$, then for $i, j \in M_1 (=M_2)$, since $parity(\alpha_i) = parity(\alpha_j)$, $parity(\alpha_i + \alpha_j) = 0$. Therefore $parity(\delta_1) = 0$. Similarly $parity(\delta_2) = parity(M_1)$ because for $j \in X_1 (=X_2)$, $parity(\alpha_j) = 0$. Clearly δ_3 is also a quarter-integer. The update rule for dual variables then makes the claim true for the new values of dual variables.

If $M_1 \neq M_2$, then some vertices of X_1 are now matched. These newly matched vertices must be connected to vertices in M_1 in the tight graph G_c. Therefore there exist $j \in M_2 \cap X_1$, and $i \in M_1$ such that the edge (i,j) is tight. Now the parities of α_j, γ_k and c_{ij} are 0. Hence $parity(\alpha_i) = 0 = parity(M_i)$. Thus δ will be a quarter-integer and the the claim will hold for the new values of the dual variables. □

So we see that at all times during the execution of the algorithm the dual variables have a defined parity and are therefore quarter-integers. Statement (b) can be proved in a fashion similar to that for bipartite graphs.

Statement (c) above suggests scaling by 4. The scaled primal (3.4A) and its dual (3.4B) appear below.

$$\min \sum_{i\,j} \left\lfloor \frac{c_{ij}}{4} \right\rfloor x_{ij}'$$

$$\sum_{j=1}^{n} x_{ij}'=1 \qquad i=1,\ldots,n \tag{3.4A}$$

$$\sum_{i\,j \in S_k} x_{ij}' + w_k' \le s_k \qquad \text{for } k=1,\ldots,N.$$

$$x_{ij}' \ge 0 \qquad w_k' \ge 0$$

$$\max \sum_{i=1}^{n} \alpha_i' + \sum_{k=1}^{N} s_k \gamma_k' \tag{3.4B}$$

$$\alpha_i' + \alpha_j' + \sum_{i\,j \in S_k} \gamma_k' \le \left\lfloor \frac{c_{ij}}{4} \right\rfloor \qquad \text{for all } i,\, j \le n$$

$$\gamma_k' \le 0 \qquad \text{for all} \qquad k \le N$$

Thus if (x_1, y_1) is a solution to (3.4A,B), then $(x_1, 4y_1)$ is a feasible solution to (3.3A,B) which is within $dx_1 \le \frac{3n}{2}$ of the optimum. At this point if we should start with $4y_1$ as an initial choice for (3.3B) and attempt to use Edmonds's algorithm, then we would run into a problem. Edmonds's algorithm needs the following invariants to be true at all stages of the algorithm.

I The x_{ij} are 0-1, and they form a matching of the graph induced by the tight edges.

II If $\gamma_k < 0$ then S_k is full, i.e., S_k has s_k matched edges.

III If $\gamma_j<0$, $\gamma_k<0$ and $S_j\cap S_k\neq\emptyset$ then $S_j\subseteq S_k$ or $S_j\supseteq S_k$.

We call a feasible solution *tight* if it obeys the above invariants. Whereas (x_1,y_1) is tight for (1.5A) and its dual, $(x_1,4y_1)$ may not be tight for (1.4A) and its dual (1.4B). Gabow's algorithm [4] circumvents this problem by modifying $(x_1,4y_1)$ to another feasible solution that is tight and also within $O(n)$ of the optimal value of the matching.

We approach the problem in another way. Now consider a solution to (3.4A,B). If in the dual solution some odd-set variables γ_i' are negative (i.e., $\gamma_i'=\gamma_i^0<0$), then the slack vector e has positive entries corresponding to the negative valued odd-set variables. Then the tuning problem (3.5A) and its dual (3.5B) are:

$$\min \sum_{i,j} c_{ij}'x_{ij}+\sum_{k=1}^{N}\gamma_k^0 y_k$$

$$\sum_{j=1}^{n}x_{ij}=1 \qquad i=1,\dots,n \tag{3.5A}$$

$$\sum_{i,j\in S_k}x_{ij}+\ y_k=s_k \qquad \text{for } k=1,\dots,N$$

$$x_{ij}\geq 0 \quad y_k\geq 0$$

$$\max \sum_{i=1}^{n}\alpha_i'+\sum_{k=1}^{N}s_k\gamma_k' \tag{3.5B}$$

$$\alpha_i'+\alpha_j'+\sum_{i,j\in S_k}\gamma_k'\leq c_{ij}' \qquad \text{for all } i,j\leq n$$

$$\gamma_k'\leq\gamma_k^0 \qquad \text{for all } k\leq N$$

Thus the format of the tuning problem is different from from the standard matching problem. The crucial difference here is that unlike the general matching LP, some odd-set variables have positive weights in the cost function. The combinatorial meaning of this is that the weighted blossoms (which were full) in the solution of (3.4A) are to be made as full as possible in the tuning problem. We shall call the above format the *tuning format* of the matching problem.

We shall adapt Edmonds's algorithm to this format of the matching problem. First extend the invariants to I',II',III' as shown below.

I' The x_{ij} are 0-1, and they form a matching of the graph induced by the tight edges.

II' If $\gamma_k' < \gamma_k^0$ then S_k is full.

III' If $\gamma_j' < \gamma_j^0, \gamma_k' < \gamma_k^0$, and $S_j \cap S_k \neq \emptyset$, then $S_j \subseteq S_k$ or $S_j \supseteq S_k$.

Consider the initial solution to (3.5B).

$$\alpha_i' = \frac{1}{2} \min_j (c_{ij} - \sum_{i,j \in S_k} \gamma_k^0)$$

$$\gamma_k' = \gamma_k^0$$

It is clear that the above solution is indeed feasible in (3.5B). We shall formulate some upper and lower bounds for the cost function evaluated on this solution.

Let the scaled down LP (3.4A) refer to the graph G', with scaled down edge costs. Suppose the problem (3.4A) has been solved recursively, hence we have a minimum cost matching of G'. These matched edges must be tight in the LP (3.4B). This implies that the slack $e_{ij}=0$ for every matched edge (i,j) in G'. This in turn implies that in (3.5B), $c_{ij}' = d_{ij} + e_{ij} = d_{ij} = c_{ij} \bmod 4 \leq 3$.

Thus

$$\sum_{i=1}^{n} \alpha_i' + \sum_{k=1}^{N} s_k \gamma_k' = \sum_{ij \text{ matched in } G'} (\alpha_i + \alpha_j + \sum_{i,j \in S_k} \gamma_k^0) \leq \frac{n}{2} \cdot 3 = \frac{3n}{2}.$$

Thus $\frac{3n}{2}$ is an upper bound on the cost of an optimal solution to (3.5B). Also note that

$$\alpha_i' = \frac{1}{2} \min_j \{c_{ij} - \sum_{i,j \in S_k} \gamma_k^0\} \geq -\frac{1}{2} (\sum_{i \in S_k} \gamma_k^0)$$

Therefore

$$\sum_{i=1}^{n}\alpha_i' + \sum_{k=1}^{N} s_k\gamma_k' \geq \frac{1}{2}\sum_{k}(-\gamma_k^0)$$

Define $\sigma = \frac{1}{2}\sum_k \gamma_k^0$. From Section 1 we know that the optimal value of the tuning problem (3.5A) is at most $\frac{3n}{2}$. The value of our initial solution to (3.5B) is at least $-\sigma$.

Next we show that Edmonds's algorithm may be extended to the tuning format of the matching problem. First define J_e and J_b as follows.

$$J_e = \{(i,j) \mid \alpha_i' + \alpha_j' + \sum_{ij \in S_k}\gamma_k' = c_{ij}'\}$$

$$J_b = \{S_k \mid \gamma_k' = \gamma_k^0\}$$

The Primal-Dual algorithm solves the tuning format of the weighted matching problem as follows: Start with the above initial dual feasible solution and then perform several iterations of solving a *restricted primal* (3.6A) stated below, and update of the dual solution accordingly. The restricted primal (3.6A) depends on the sets J_e, J_b above.

$$\min \sum_{j=1}^{n} x_j^a + 2\sum_{k=1}^{N} x_{n+k}^a$$

$$\sum_{j=1}^{n} x_{ij} + x_i^a = 1 \qquad i = 1,\ldots,n \qquad (3.6A)$$

$$\sum_{i,j \in S_k} x_{ij} + w_k + x_{n+k}^a = s_k \qquad k = 1,\ldots,N$$

$$x_{ij} \geq 0 \qquad (i,j) \text{ not} \in J_e \rightarrow x_{ij} = 0$$

$$w_k \geq 0 \qquad S_k \text{ not} \in J_b \rightarrow y_k = 0$$

The dual of (3.6A) is (3.6B) below.

$$\max \sum_{i=1}^{n}\alpha_i + \sum_{k=1}^{N} s_k\gamma_k$$

$$\alpha_i + \alpha_j + \sum_{i,j \in S_k}\gamma_k \leq 0 \quad (i,j) \in J_e \qquad (3.6B)$$

$$\gamma_k \leq 0$$

$$\alpha_i \leq 1 \quad \text{for all } i, \quad \gamma_k \leq 2 \quad \text{for all } k$$

Note that the restricted primal and dual problems for the standard format and the tuning format are identical. Under the modified invariants I',II' and III', Edmonds's algorithm may be used to obtain an optimal solution to (3.5A) and (3.5B). The next question is the time analysis of the above procedure on (3.5A),(3.5B). First note that the cost of our initial solution to (3.5B) is at least $-\sigma$. After the trimming procedure of Section 1 each entry in

$$\sum_{i,j} c_{ij}' x_{ij} + \sum_{k=1}^{N} \gamma_k^0 y_k,$$

the cost vector of (3.5A) can be made less than $\frac{3n}{2}+2$. Further, since the γ_k^0's were obtained from the previous scale, the odd-sets corresponding to these variables represent a valid blossom structure. This implies that the number of non-zero γ_k^0 equals the number of blossoms in the previous scale and thus is bounded by $\frac{n}{2}$. As each γ_k^0 is less than $\frac{3n}{2}+2$, σ is $O(n^2)$. But the optimal value of (3.5B) is at most $\frac{3n}{2}$. Thus our initial solution is $O(n^2)$ away from the optimum.

Next we attempt to bound the number of dual variable updates, or steps, required by the modified Edmonds's algorithm starting with the stated initial solution. First note that $\theta = \min\{\delta_1, \delta_2, \delta_3\}$ is at least $\frac{1}{4}$. If at any stage during the execution of the algorithm the number of exposed vertices is f, then the value of the cost function increases by θf in that step, which is at least $\frac{f}{4}$.

Initially all vertices are exposed and finally none remain exposed. Let k_i be the number of steps between the ith and the $(i+1)$st augmentation. Then $k_i \leq n$ for all i (See [7], Section 11.3). Each augmentation reduces the number of exposed vertices by 2. Furthermore, we know that the total increase in the cost function is $O(n^2)$. Since the ith augmentation increases the value of the objective function by $n-2i$, finding an upper bound on the total number of steps reduces to the follwing problem.

$$\max k_1+k_2+\ldots+k_n$$

$$\sum_{i=1}^{n} k_i(n-2i)=cn^2$$

$$k_i \le n \text{ and } k_i \ge 0 \text{ for all } i.$$

The optimal value of the above problem is easily seen to be $O(n^{1.5})$, which is better than $O(n^2)$, our previous estimate. This is going to reduce the number of steps required for Edmonds's algorithm.

CONCLUSIONS

The scaling approach has been widely used to design efficient algorithms for specific problems. The highlight of this paper was to examine scaling from a more general framework of linear programs (LPs) and to present a unified picture of scaling. Thus scaling may be seen as another approach to solving LPs. We showed that the typical scaling algorithm depends, for its efficiency, on (what we call) the tuning problem. The tuning problem (TP) is the problem of translating a solution to the previous scale into a solution to the current scale, and it is the only problem specific part of the algorithm. The TP, although similar in format to the original problem, is much simpler.

Scaling fails on the tuning problem. Most algorithms use problem-specific approaches that exploit the structure of the TP. Gabow and Tarjan proposed an approximation algorithm for the TP that arises from the general weighted matching problem. We generalised this approach to some extent and showed its relation to the Primal-Dual algorithm. The analysis of the above approach is very tedious, and we feel that a more general framework is required.

Another approach to the solution of LPs was Bertsekas's relaxed complementary slackness. We showed that this may be seen as the dual of the scaling approach, i.e., scaling of the primal problem is equivalent to Bertsekas's approach to the dual problem.

REFERENCES

[1] D. P. BERTSEKAS, Distributed Relaxation Methods for Linear Network Flow Problems, *Proc. IEEE Conf. Decision* and *Control*, Athens, Greece, 1986, pp. 2101-2106.

[2] J. EDMONDS and R. M. KARP, Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems, *JACM*, vol. 19, no. 2, 1972, pp. 248-264.

[3] H. N. GABOW, Scaling Algorithms for Network Problems, *J. Comput. Syst. Sci.*, vol. 31, 1985, pp. 148-168.

[4] H. N. GABOW, A Scaling Algorithm for Weighted Matchings on General Graphs, *Proc. 26th Ann. Symp. on Foundations of Computer Science*, Portland, Oregon, 1985, pp. 90-100.

[5] H. N. GABOW and R. E. TARJAN, Faster Scaling Algorithms for Network Problems, *Tech. Report* CS-TR-111-87, Aug. 1987, Department of Computer Science, Princeton University.

[6] A. V. GOLDBERG and R. E. TARJAN, Solving Minimum Cost Flow Problems by Succesive Approximation, *Proc. 19th Ann. ACM Symp. on Theory of Computing*, New York, NY, 1987, pp. 7-18.

[7] C. H. PAPADIMITRIOU and K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, 1982.